

## Group 12A

Christoffer Staupe, [christofns@uia.no](mailto:christofns@uia.no)

Truls Iver M. Iversen, [tiiversen@uia.no](mailto:tiiversen@uia.no)

Lene Olsen, [lenco12@uia.no](mailto:lenco12@uia.no)

Isak Salvesen Galleberg, [isaksg@uia.no](mailto:isaksg@uia.no)

Marius Horni, [mariushor@uia.no](mailto:mariushor@uia.no)

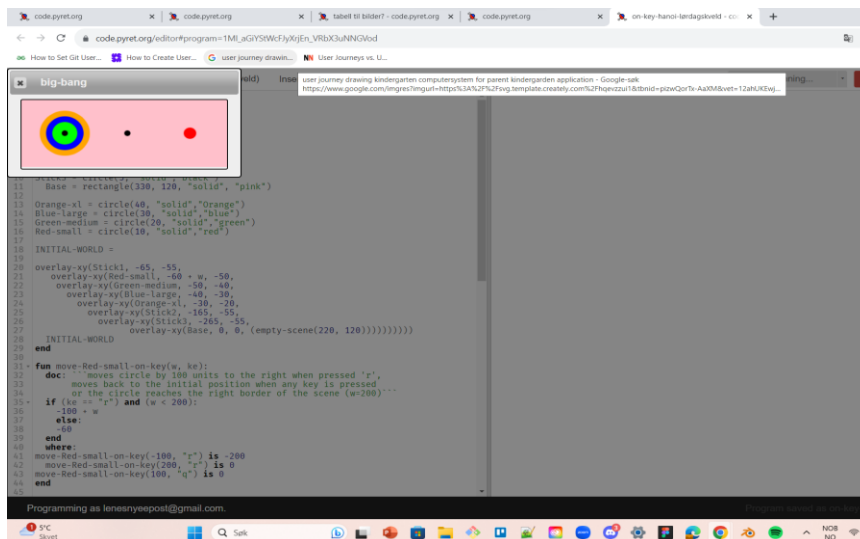
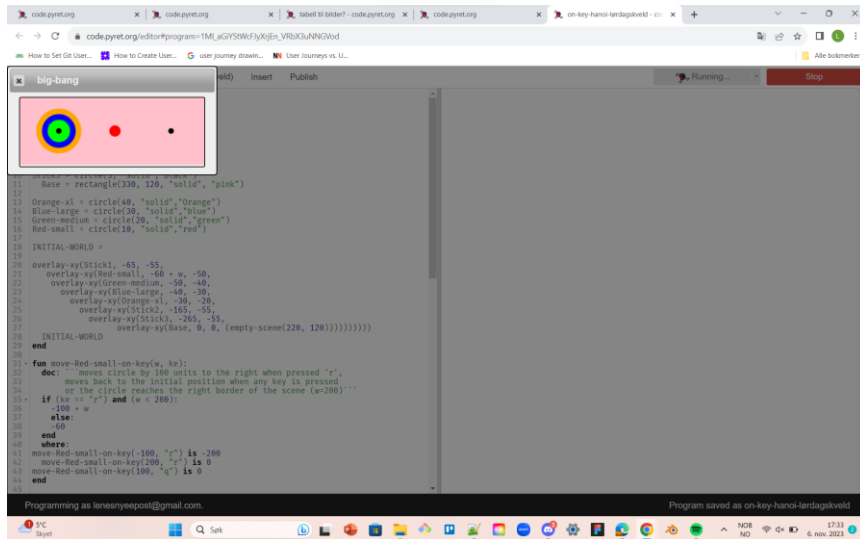
Alexandros Dimopoulos, [alexandrod@uia.no](mailto:alexandrod@uia.no)

|                                |   |
|--------------------------------|---|
| <b>Course Code &amp; Name:</b> | IS-114 Introduksjon til samskaping I informasjonssystemer |
| <b>Deliverable Title:</b>      | Progoblig 02 (Gruppe Del)                                 |
| <b>Date:</b>                   | 06.11.2023  |

|  |       |    |
|--|-------|----|
| We confirm hereby that we are not citing or using in other ways other's work without stating that clearly and that all of the sources that we have used are listed in the references | Yes X | No |
| We allow the use of this work for educational purposes   | Yes X | No |
| We hereby confirm that every member of the group named on this page has contributed to this deliverable/product  | Yes X | No |

## Progoblig 02 (Gruppe Del)

Vi startet oppgaven veldig ambisiøst og gikk rett inn og brukte pakken “World” som inneholdt funksjonen “Big Bang”, hvor man flyttet på sirklene med hjelp av tastetrykk. Som gruppe merket vi fort at det ble store problemer å løse med denne metoden og ble sittende fast på flere stadier. Problemer vi støttet på var blant annet: fikk bare en sirkel til å flytte seg, kun scenen spillet var lagt inn i flyttet seg ...



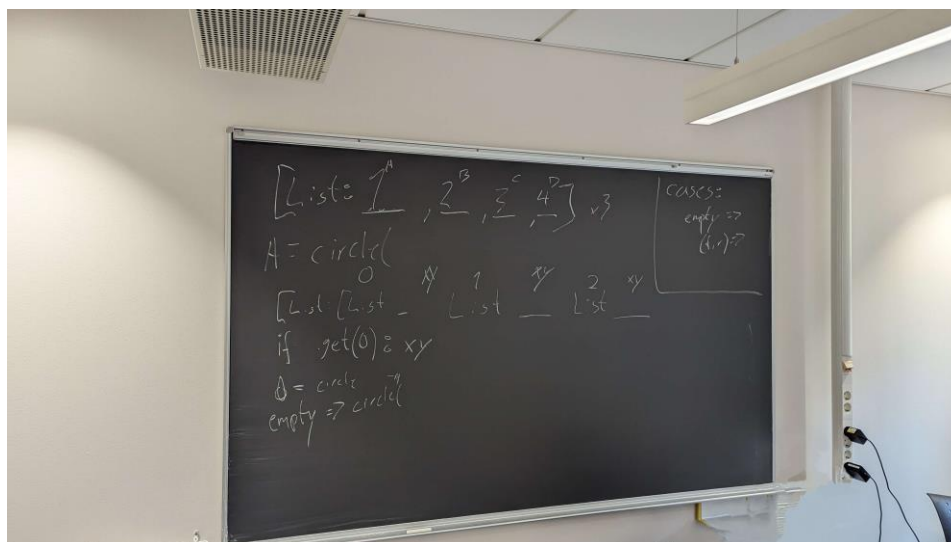
Etter besøk av foreleser ende vi opp med å gå for "list"-funksjon, ved å legge tre lister (en for hvert hull) i en liste, for å så lage en funksjon som gjorde at vi kunne flytte verdiene mellom de ulike listene for å simulere spillet. Vi så dette som en akseptabel metode ettersom funksjonen har en likhet med spillets regler, i form av at man bare kan flytte på den øverste verdien/sirkel.

```

1 use context essentials2021
2 tilstand = [list: [1, 2, 3, 4], [list: 0, 0, 0, 0], [list: 0, 0, 0, 0]]
3
4 den liste som inneholder liste nr 0, 1 og 2
5
6
7 fun move(fr, to, tilst):
8     if (fr < 2) and (to < 2):
9         pinne1 = tilst.get(0)
10        pinne2 = tilst.get(1)
11        pinne3 = tilst.get(2)
12        trek = pinne1.set(0, fr) #move øverste element "fr" til pinne "to" ut fra
13        tilstand list:
14            trek = pinne2.set(0, 2)
15            fase2 = tilst.set(0, pinne1.set(0, 0))
16            fase2.set(1, pinne2.set(0, 1))
17        else:
18            raise("ulovlig trekk")
19        end
20
21 tilstand
22
23
24
25 move(1, 2, tilstand)

```

Brainstorming for å kunne komme frem til vidrere løsninger for å bruke liste funksjonen til å løse spillet.



Bildet under er en tabell laget med tanke på at man kunne flytte elementene i tabellen, for å så spille spillet.

|    | A             | B      | C      | D | E | F | G | H | I | J | K | L | M |
|----|---------------|--------|--------|---|---|---|---|---|---|---|---|---|---|
| 1  | Stick1        | Stick2 | Stick3 |   |   |   |   |   |   |   |   |   |   |
| 2  | *red-small    |        |        |   |   |   |   |   |   |   |   |   |   |
| 3  | *green-medium |        |        |   |   |   |   |   |   |   |   |   |   |
| 4  | *blue-large   |        |        |   |   |   |   |   |   |   |   |   |   |
| 5  | *orange-all   |        |        |   |   |   |   |   |   |   |   |   |   |
| 6  | *base         | *base  | *base  |   |   |   |   |   |   |   |   |   |   |
| 7  |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 8  |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 9  |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 10 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 11 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 12 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 13 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 14 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 15 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 16 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 17 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 18 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 19 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 20 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 21 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 22 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 23 |               |        |        |   |   |   |   |   |   |   |   |   |   |
| 24 |               |        |        |   |   |   |   |   |   |   |   |   |   |

De fire neste bildene illustrerer våre forsøk på å visuelt kunne vise frem trekkene.

Når vi velger å bruke overlay sammen med en empty-scene med 3 pinner/sorte sirkler legger liste 1 (alle ringene) seg under første sorte sirkel (pinne nr.

1 til venstre), så da er ideen at vi kan bruke samme empty-scene, men med færre pinner i koden på rett punkt så vil illustrasjonen av listen legge seg på den pinnen som er over i koden. Dette gjør den da ikke, og ringen ligger skjult under de andre ringene. Vi kan illustrere at ringen er synlig og ligger på riktig plass i riktig liste om vi kaller frem illustrasjon av kun det liste punktet.

Om vi navngir denne liste posisjonen sammen med en empty-scene, klarer vi sette den sammen med listen som nå har kun 3 ringe og dens empty-scene og få en visuell fremstilling av at vi har flyttet rød-sirkel til pinnen helt til høyre.

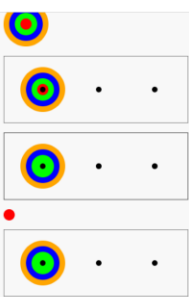
code.pynet.org/ide/editor#program=186w1T7X42Hf0d600uApC9C\_Dmg\_VH...

How to Set Git User... How to Create User... user journey draw... User Journeys vs U...

View File (hanoi 06.11. 17.38) Insert Publish

```
71 hanoi-board-new-state-2
72 #overlay(overlay-xy(Stick1, -65, -55,
73   overlay-xy(Stick2, -165, -55,
74     overlay-xy(Stick3, -265, -55,
75       overlay-xy(red-small, -60, -50,
76         overlay-xy(green-medium, -50, -40,
77           overlay-xy(blue-large, -40, -30,
78             overlay-xy(orange-xl, -30, -20,
79               (base)))))))))#
80
81 #overlay(overlay(overlay(overlay(circle(5, "solid", "black", (hanoi-
82   board.get(0).get(0), hanoi-board.get(0).get(1)), hanoi-board.get(0).get(2)), hanoi-
83     board.get(0).get(3), empty-board))))))#
84
85 overlay(overlay(overlay(hanoi-board.get(0).get(0), hanoi-board.get(0).get(1)),
86   hanoi-board.get(0).get(2)), hanoi-board.get(0).get(3))
87
88 overlay(overlay(overlay(hanoi-board.get(0).get(0), hanoi-board.get(0).get(1)),
89   hanoi-board.get(0).get(2)), hanoi-board.get(0).get(3))
90
91 overlay(overlay(overlay(overlay(empty-board,
92   hanoi-board-new-state-2.get(0).get(0)),
93     hanoi-board-new-state-2.get(0).get(1)),
94       hanoi-board-new-state-2.get(0).get(2)),
95         empty-board),
96           hanoi-board-new-state-2.get(2).get(0))
97
98 hanoi-board-new-state-2.get(2).get(0)
99
100 overlay(overlay(overlay(overlay(empty-board,
101   hanoi-board-new-state-2.get(0).get(0)),
102     hanoi-board-new-state-2.get(0).get(1)),
103       hanoi-board-new-state-2.get(0).get(2)),
104         hanoi-board-new-state-2.get(2).get(0))
105
106
107
108 #!fun draw-overlay(acc, elt) -> Image:
109   overlay(acc, elt)
110
111 endfun
```

Run Stop



Looks shipshape, your test passed, mate!

check-block-1  
The test in this block passed

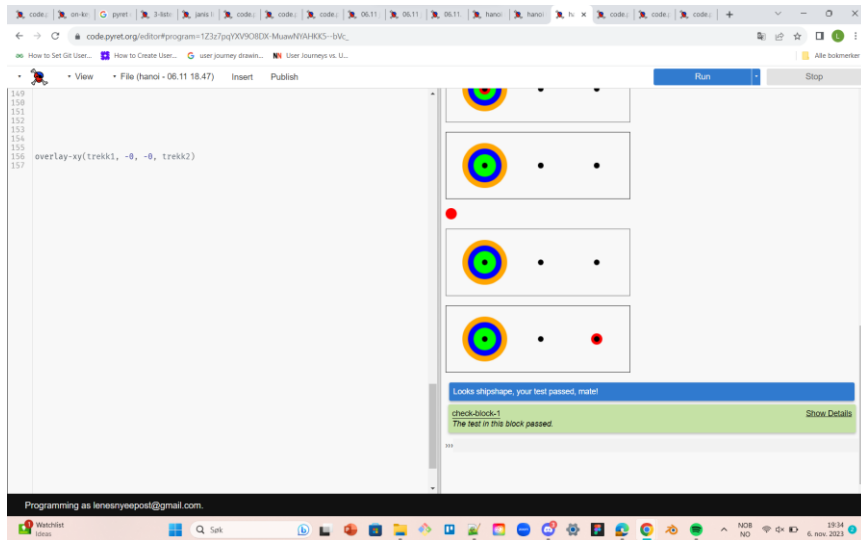
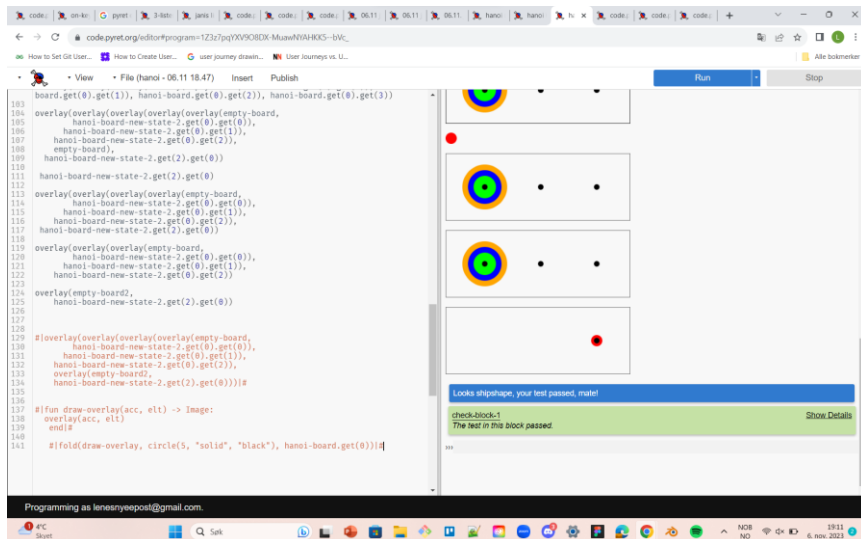
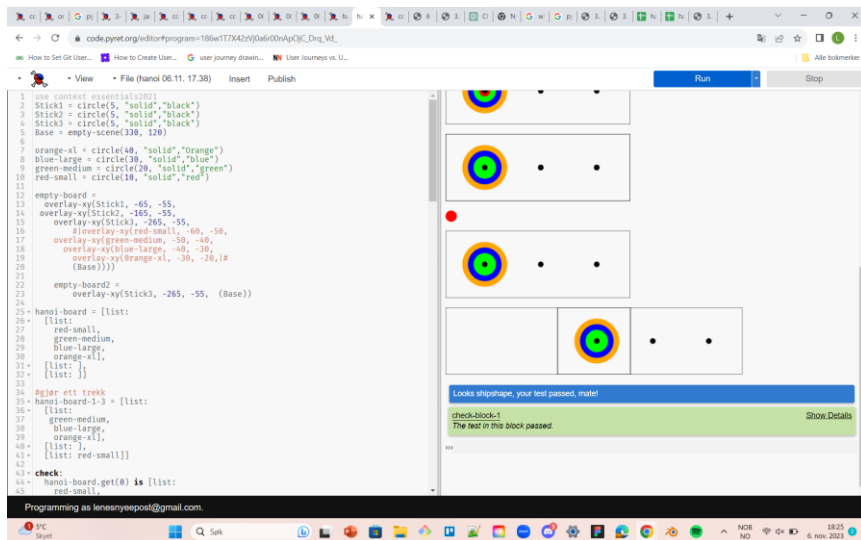
Show Details

111

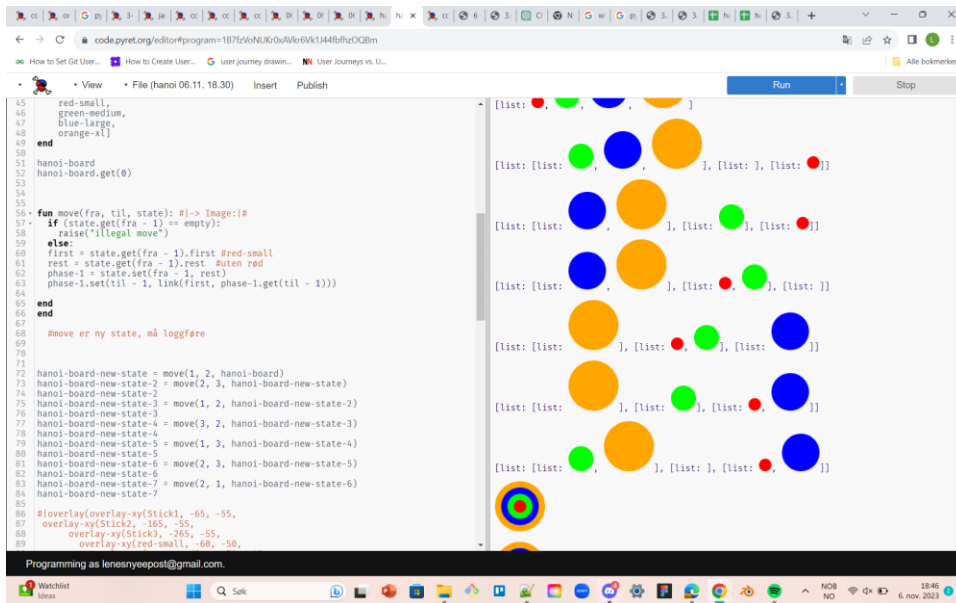
Programming as lenesyeepost@gmail.com.

10C Skyer Q Sek

1816 6. nov. 2023



I bildet under kan vi vise at vi får gjort trekk, med koden, og de forskjellige trekkene er lagret med forskjellige navn.



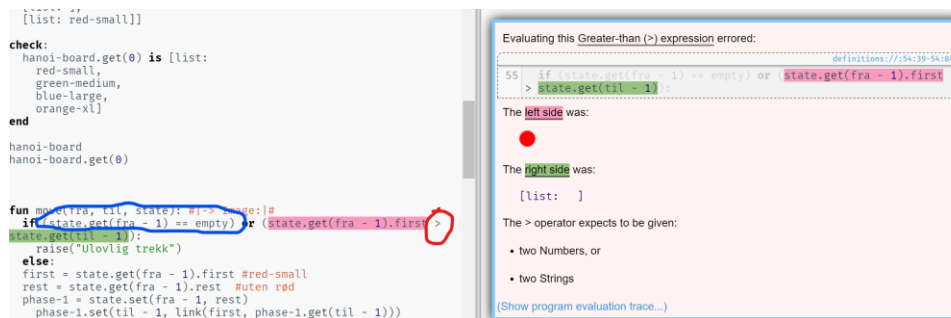
```
69
70
71
72 hanoi-board-new-state = move(1, 2, hanoi-board)
73 hanoi-board-new-state-2 = move(2, 3, hanoi-board-new-state)
74 hanoi-board-new-state-2
75 hanoi-board-new-state-3 = move(1, 2, hanoi-board-new-state-2)
76 hanoi-board-new-state-3
77 hanoi-board-new-state-4 = move(3, 2, hanoi-board-new-state-3)
78 hanoi-board-new-state-4
79 hanoi-board-new-state-5 = move(1, 3, hanoi-board-new-state-4)
80 hanoi-board-new-state-5
81 hanoi-board-new-state-6 = move(2, 3, hanoi-board-new-state-5)
82 hanoi-board-new-state-6
83 hanoi-board-new-state-7 = move(2, 1, hanoi-board-new-state-6)
84 hanoi-board-new-state-7
85
```

**Forsøk på å definere ‘ulovlige trekk’:**

```

fun move(fra, til, state): #|-> Image:|#
  if (state.get(fra - 1) == empty) or (x > y)
    raise("Ulovlig trekk")
  else:
    first = state.get(fra - 1).first #red-small
    rest = state.get(fra - 1).rest #uten rød
    phase-1 = state.set(fra - 1, rest)
    phase-1.set(til - 1, link(first, phase-1.get(til - 1)))

```



Bildene ovenfor viser forsøk på å sette inn “If-else” + feilmelding for å varsle om “Ulovlige trekk”. Hvis ‘input-ene’ ikke tilfredsstiller kravene, vil ikke et trekk gå gjennom, og spilleren blir heller møtt med en feilmelding. Den første feilmeldingen (markert med blått) skal hindre spilleren i å forsøke å flytte en brikke fra en posisjon hvor det ikke er en brikke å flytte.

Den andre feilmeldingen (markert med rødt) fikk vi ikke til, men skulle hindre spilleren i å legge en større brikke/sirkel over på en mindre sirkel: ved å sjekke om brikken man skulle til å flytte hadde en større verdi (i dette tilfellet radius) enn brikken som lå først i posisjonen man skulle flytte til, kunne man gjenkjenne et ulovlig trekk. Fikk det ikke helt til å gå når funksjonen ble matet med en tom liste :/. Kanskje ved å sette inn flere kondisjonaler, som at tom liste (som har mindre verdi enn sirkel) vil være ok eller noe sånt - ikke sikker.

**Inkludering av ‘tegne-funksjonen’ i ‘move-funksjonen’:**



```

hanoi-board
hanoi-board.get(0)

fun move(fra, til, state): #!-> Image:#
if (state.get(fra - 1) == empty) or (state.get(fra - 1).first >
state.get(til - 1)):
    raise("Ulovlig trekk")
else:
    first = state.get(fra - 1).first #red-small
    rest = state.get(fra - 1).rest #uten rød
    phase-1 = state.set(fra - 1, rest)
    phase-1.set(til - 1, link(first, phase-1.get(til - 1)))

    # overlay(overlay(overlay(hanoi-board.get(0).get(0), hanoi-
board.get(0).get(1)), hanoi-board.get(0).get(2)), hanoi-
board.get(0).get(3))
    # ->
    overlay(overlay(overlay(state.get(0).get(0),
state.get(0).get(1)), state.get(0).get(2)), state.get(0).get(3))
    # Vil kun vise 'liste 1'. For å få alle listene og sirklene
    # kunne man kanskje ha lagt et par 'overlay-koder' til for
    # å inkludere .get(1) og .get(2)

end
end

#move er nv state. må løseføre

```

This expression contains a block that contains multiple expressions.

```

55 if (state.get(fra - 1) == empty) or (state.get(fra - 1).first
> state.get(til - 1)):
56     raise("Ulovlig trekk")
57 else:
58     first = state.get(fra - 1).first #red-small
59     rest = state.get(fra - 1).rest #uten rød
60     phase-1 = state.set(fra - 1, rest)
61     phase-1.set(til - 1, link(first, phase-1.get(til - 1)))
62
63     # overlay(overlay(overlay(hanoi-board.get(0).get(0),
hanoi-board.get(0).get(1)), hanoi-board.get(0).get(2)), hanoi-
board.get(0).get(3))
64     # ->
65     overlay(overlay(overlay(state.get(0).get(0),
state.get(0).get(1)), state.get(0).get(2)), state.get(0).get(3))
66     # Vil kun vise 'liste 1'. For å få alle listene og sirklene
67     # kunne man kanskje ha lagt et par 'overlay-koder' til for
68     # å inkludere .get(1) og .get(2)
69
70
71 end

```

Either simplify this block to a single expression, or mark the outer expression with block: to indicate this is deliberate.

Bildet ovenfor viser et forsøk på å inkludere ‘tegne-funksjonen’ (‘overlay’) inn i ‘move-funksjonen’, slik at ‘state’-en som blir tegnet samsvarer med trekkene man tar – fikk ikke dette helt til :0. (Burde kanskje bruke to funksjoner siden en funksjon godtar ikke 2 uttrykk – men ble for øyeblikket ikke klok på hvordan to funksjoner kunne bruke samme variabel (‘state’).

## Angre-funksjon:

```

fun move(fra, til, state):
    if (state.get(fra - 1) == empty):
        raise("illegal move")
    else:
        first = state.get(fra - 1).first
        rest = state.get(fra - 1).rest
        phase-1 = state.set(fra - 1, rest)
        phase-1.set(til - 1, link(first, phase-1.get(til - 1)))

end
end

```

```

hanoi-board-new-state = move(1, 2, hanoi-board)
hanoi-board-new-state-2 = move(2, 3, hanoi-board-new-state)
hanoi-board-new-state-3 = move(1, 2, hanoi-board-new-state-2)
hanoi-board-new-state-4 = move(1, 2, hanoi-board-new-state-3)
hanoi-board-new-state-4

```

For å reversere trekk, var tanken å ha en form for ‘logg’ hvor hvert trekk var lagret, slik at man kunne få tilgang til en tidligere tilstand: reversere trekk = flytte en tilstand tilbake. Med den (kanskje tungvinte) måten trekk/’moves’ blir gjort med denne funksjonen vil tidligere tilstander

være lagret (ettersom man lagrer hver tilstand), og med det kan 'fiske' ut hvilket som helst trekk man har gjort.

En idé (som vi ikke har produsert), ville vært å ha en 'knapp' eller liknende for å kjapt reversere et trekk. En annen idé var å gjøre logg-føring mer 'smooth', ved at det kanskje automatisk blir lagret i en tabell eller liknende, og i stedet for å skrive en ny tilstand hver gang, så kunne man bygge på en tidligere tilstand og på en måte bare legge til '1' (som i trekk 1 -> trekk 2).

Kunne også tenkt å legge til en 'restart'-funksjon ('loade' initial-tilstand).

Koden ligger i docs prog.oblig 02 i denne repositoryen <https://github.com/OlsenLene/4school.git>